浙江大学

Matlab 图 像 处 理 编 程 实 践
第 二 次 大 作 业 报 告

课程名称：	Matlab图像处理
姓　　名：	王晨雨
学　　院：	计算机科学与技术学院
系：	计算机科学与技术
专　　业：	计算机科学与技术
学　　号：	3200102324

2022 年 7 月 11 日

目 录

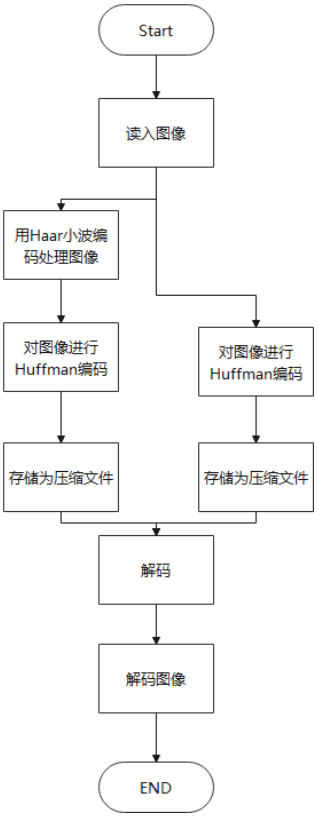
1. 实验任务简介

本次实验要求我们利用 Haar 小波编码，得到中间数据文件，存储。针对编码后的中间存储文件，利用 matlab 内嵌的 huffman 编码函数进行二进制编码，并存为压缩文件；读取压缩文件，解码得到原始图像进行显示并对比压缩效率。

2. 程序框架与技术细节

2.1 程序框架展示

总体程序框架（流程图展示）：



main.m：

```
I = imread('m5.png');
I = imresize(I, [256, 256]);
I_Huffman = Huffman(I);
save('I_Huffman.mat',"I_Huffman");
I_Haar_Huffman = Huffman(Haar(I));
save('I_Haar_Huffman.mat', "I_Haar_Huffman");
subplot(1, 3, 1);
imshow(I);
xlabel('(a)原图');
```

```
subplot(1, 3, 2);
imshow(Huffman_de(I));
xlabel('(b)原图 Huffman 压缩');
subplot(1, 3, 3);
imshow(Huffman_de(Haar(I)));
xlabel('(c)Haar 小波变换后 Huffman 压缩');
```

main.m 为最上层的部分。通过调用 Haar/Huffman_de/Huffman 函数来展示原图像和解码后的图像。

Huffman.m：

Huffman.m 将矢量图像转换为 Huffman 编码并且存在 result 中。

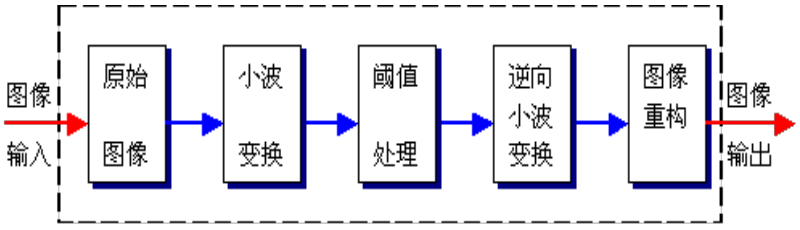
result 传递到上层的 main.m

```
function [result] = Huffman(image)
[m, n] = size(image);
vector = image(:);
p = zeros(1, 256);
for i = 1:256
    p(i) = length(find(vector==(i-1))) / (m*n);
end
k = 0:255;
dict = huffmandict(k, p);
enco = huffmanenco(vector, dict);
result = enco;
end
```

Haar.m: 将图像进行 Haar 小波变换，经小波变换后的图像具有更宽的范围，但是数据的动态范围缩小，实现图像的高效压缩。

```
function[w] = Haar(a)
```

基于小波变换的图像压缩的基本流程:



通过行列变换后得到矩阵A_{RC}，将A_{RC}进行阈值裁剪：设定阈值为 5:

```
for i = 2:256
    if res(i) < 5 && res(i) > -5
        res(i) = 0;
    end
end
```

返回矩阵 w 为小波变换后的结果。

Huffman_de: 将 Huffman 编码后的图像解码，将解码后的图像返回到 main.m.

最后在 main.m 里面输出三张图片。分别为原图、原图经过 Huffman 压缩后解码得到的图像、原图先 Haar 小波变换后再 Huffman 压缩后解码得到的图像。

2.2 实现细节：

1. 注意统一图片大小，main.m 里将图片先通过 imresize 函数转换为[256, 256]大小。
2. Huffman 压缩后文件以.mat 形式存储。
3. 为了对比压缩效率，我把 lab4 的代码改了一下，用差分矩阵经过 huffman 压缩，即文件 diff_encode_Huffman.m 中的代码。

3. 程序运行示例

在 I = imread();中输入您想处理的图像名，我在文件夹里提供了三张测试图像分别为 test1.png, test2.png, test3.png, 下图为 test3.png 的测试代码。我在第四部分测试结果分析时也将三张图片都测试了一遍。

```
I = imread('test3.png');
I = imresize(I, [256, 256]);
I_Huffman = Huffman(I);
save('I_Huffman.mat',"I_Huffman");
I_Haar_Huffman = Huffman(Haar(I));
save('I_Haar_Huffman.mat', "I_Haar_Huffman");
subplot(1, 3, 1);
imshow(I);
xlabel('(a)原图');
subplot(1, 3, 2);
imshow(Huffman_de(I));
xlabel('(b)原图Huffman压缩');
subplot(1, 3, 3);
imshow(Huffman_de(Haar(I)));
xlabel('(c)Haar小波变换后Huffman压缩');
```

打开 main.m，运行，输出三张图片。如下图所示：分别为原图、原图经过 Huffman 压缩后解码得到的图像、原图先 Haar 小波变换后再 Huffman 压缩后解码得到的图像。

另外，diff_encode_Huffman.m 为我 lab4 的代码的修改版本，即通过差分矩阵再进行 Huffman 压缩，encode 里即为 Huffman 压缩后的结果，您可以根据 encode 和 I（原始图像）的大小来计算压缩效率。您可以直接运行 diff_encode_Huffman.m 即可。

4. 实验结果分析

4.1 实验结果展示

1. 读入图片为 test2.png. 点击 main.m 运行。输出结果为:



(a)原图　(b)原图Huffman压缩　(c)Haar小波变换后Huffman压缩

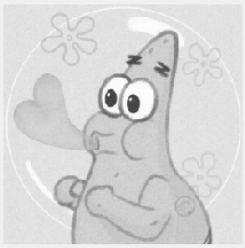
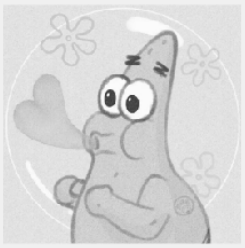
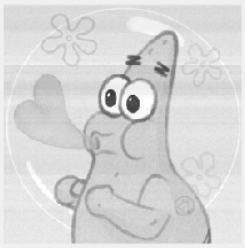
工作区变量的结果:

名称	值
I	256x256 uint8
I_Haar_Huffman	491483x1 double
I_Huffman	489801x1 double

可以计算得:
原图大小: $256 \times 256 \times 8 = 524288\ bits$
未经 Haar 小波变换: 489801 $bits$, 压缩效率: 93.5%
经 Haar 小波变化: 491483 $bits$, 压缩效率: 93.4%

运行 diff_encode_Huffman.m，比较经过差分编码的压缩效率：93.4%

2. 读入图片为 test1.png. 点击 main.m 运行。输出结果为:

(a)原图　(b)原图Huffman压缩　(c)Haar小波变换后Huffman压缩

工作区变量的结果:

工作区

名称	值
I	256x256 uint8
I_Haar_Huffman	398585x1 double
I_Huffman	367709x1 double

可以计算得:
原图大小：$256 \times 256 \times 8 = 524288\ bits$
未经 Haar 小波变换：$367709\ bits$，压缩效率：70.1%
经 Haar 小波变化：$398585\ bits$，压缩效率：76.0%

运行 diff_encode_Huffman.m，比较经过差分编码的压缩效率：76.3%

3. 读入图片为 test3.png. 点击 main.m 运行。输出结果为:

工作区变量的结果:

可以计算得:
原图大小：$256 \times 256 \times 8 = 524288\ bits$
未经 Haar 小波变换：$420862\ bits$，压缩效率：81.4%
经 Haar 小波变化：$426893\ bits$，压缩效率：80.3%

运行 diff_encode_Huffman.m，比较经过差分编码的压缩效率：80.3%

4.2 结果分析：

1. 压缩效率与 haar 小波变换过程中阈值的选择有关。本次 haar 小波变换阈值设置为了 5，在压缩效率与压缩后图像质量之间追求平衡。
2. 通过还原后的图片来看，haar 小波变换由于阈值设置的问题，还原效果相比较不经过 haar 小波变换会差一点。

5. 总结与思考

通过本次实验，我学习了几种将图像压缩的方法，对比了它们之间的不同之处，并且深刻理解了这些方法（比如差分法、haar 小波变换）的原理。